模块概念

- 模块是一个包含Python定义和语句的文件，把相关的代码分配到一个模块里，可以让代码更好用、更易懂。

模块导入

- 模块可以被别的程序引入，以使用该模块中定义的变量，函数等功能
- 习惯上（但不强制要求）把所有导入语句放在模块的开头
- 一个模块被另一个程序第一次导入时，会执行该模块

模块导入
import module
import module as alias
from module import item
from module import item as alias
from module import *

注意：请慎用 from module import * ，很容易出现名称重复的情况，导致出现一些意外的问题

包的概念

- Python包本质上是一个文件夹，只是该文件夹会包含 __init__.py 模块
- 和文件夹一样，包里面还可以存在子包，模块或者其它文件

包的作用

- 避免相同命名冲突： 如果在同一个包里，是不允许两个模块命名相同的，但是如果不在同一个包里，是可以的
- 模块分区： 把不同功能的模块归类到不同的包里，方便查询和修改。在比较大型的项目中常常需要编写大量的模块，此时我们可以使用包来对这些模块进行管理

包的导入

包的导入
import package
import package as alias
from package import module
from package import module as alias

包的导入

from package.module import item

from package.module import item as alias

搜索路径

- sys模块的path变量以列表的形式记录了Python解释器自动查找所需模块或包的路径；如果这些路径都找不到，则会报错：ModuleNotFoundError: No module named 'xxx'

```
import sys



print(sys.path)
```

__name__属性

- 每个模块都有一个__name__属性，当其值是'__main__'时，说明该模块自身在运行，否则说明该模块因为被导入才执行的，此时其值为模块名
- 在完成一个模块的编写之前，我们一般会对模块中的功能进行测试，看看各项功能是否正常运行。对于这些测试的代码，我们希望

只在直接运行这个模块时执行，而在其它程序导入这个模块时不要执行，这个时候就可以借助__name__属性来实现。

```
def add(x, y):
    print(x + y)
    print(x * 2)


if __name__ == '__main__':
    add(3, 4)
    add('3', '4')
```